Assignment 8-10 Integer Division 4

The problem is to compute the quotient q and the remainder r of a positive integer a divided by a positive integer b. ($b \le a < 10^{250}$)

Input

The input consists of t ($30 \le t \le 40$) test cases. The first line of the input contains only positive integer t. Then t test cases follow. Each test case consists of two lines which give the two positive integers a and b ($b \le a < 10^{250}$), respectively.

Output

For each test case, you are to output exactly two lines containing, the quotient q and the remainder r, respectively.

Sample Input

```
3
12345
12311
12345
12345
12345
1
```

Sample Output

```
1
34
1
0
12345
0
```

Part of the program

You are required to write the function `division`, `subtraction` and `less` to complete the following program which solves this problem. In your program, you cannot declare global variables or static arrays.

```
#include<iostream>
#include<cstring>
using std::cin;
```

```cpp
using std::cout;
using std::endl;

class HugeInt
{
public:
   int size;
   int *digit;

   // quotient = dividend / divisor; remainder = dividend % divisor
   // provided that dividend != 0, divisor != 0 and dividend >= divisor
   void division( HugeInt divisor, HugeInt &quotient, HugeInt &remainder );

   // hugeInt /= 10, or equivalently, shifts right by one position
   void divideBy10();

   // minuend -= subtrahend
   // provided that minuend != 0, subtrahend != 0 and minuend >= subtrahend
   void subtraction( HugeInt subtrahend );

   // returns true if and only if hugeInt1 < hugeInt2
   // provided that hugeInt1 != 0 and hugeInt2 != 0
   bool less( HugeInt hugeInt2 );

   // return true if and only if hugeInt1 == hugeInt2
   // provided that hugeInt1 != 0 and hugeInt2 != 0
   bool equal( HugeInt hugeInt2 );

   // returns true if and only if the specified huge integer is zero
   bool isZero();
};

const int arraySize = 250;

int main()
{
   char strA[ 251 ], strB[ 251 ];

   int T;
   cin >> T;
   for( int t = 0; t < T; ++t )
   {
      cin >> strA >> strB;

      HugeInt dividend;
      dividend.size = strlen( strA );
      dividend.digit = new int[ dividend.size ]();
      for( int i = 0; i < dividend.size; ++i )
         dividend.digit[ i ] = strA[ dividend.size - 1 - i ] - '0';

      HugeInt divisor;
      divisor.size = strlen( strB );
      divisor.digit = new int[ divisor.size ]();
      for( int i = 0; i < divisor.size; ++i )
         divisor.digit[ i ] = strB[ divisor.size - 1 - i ] - '0';

      HugeInt quotient;
      HugeInt remainder;
      dividend.division( divisor, quotient, remainder );

      for( int i = quotient.size - 1; i >= 0; i-- )
         cout << quotient.digit[ i ];
      cout << endl;

      for( int i = remainder.size - 1; i >= 0; i-- )
         cout << remainder.digit[ i ];
      cout << endl;

      delete[] dividend.digit;
      delete[] divisor.digit;
      delete[] quotient.digit;
      delete[] remainder.digit;
   }
}

// quotient = dividend / divisor; remainder = dividend % divisor
// provided that dividend != 0, divisor != 0 and dividend >= divisor
void HugeInt::division( HugeInt divisor, HugeInt &quotient, HugeInt
&remainder )
{






}

// hugeInt /= 10, or equivalently, shifts right by one position
void HugeInt::divideBy10()
{
   if( size == 1 )
      digit[ 0 ] = 0;
   else
   {
      for( int i = 1; i < size; i++ )
         digit[ i - 1 ] = digit[ i ];

      size--;
      digit[ size ] = 0;
   }
}

// minuend -= subtrahend
// provided that minuend != 0, subtrahend != 0 and minuend >= subtrahend
void HugeInt::subtraction( HugeInt subtrahend )
{






}

// returns true if and only if hugeInt1 < hugeInt2
// provided that hugeInt1 != 0 and hugeInt2 != 0
bool HugeInt::less( HugeInt hugeInt2 )
{






}

// return true if and only if hugeInt1 == hugeInt2
// provided that hugeInt1 != 0 and hugeInt2 != 0
bool HugeInt::equal( HugeInt hugeInt2 )
{
   if( size != hugeInt2.size )
      return false;

   for( int i = size - 1; i >= 0; i-- )
      if( digit[ i ] != hugeInt2.digit[ i ] )
         return false;

   return true;
}

// returns true if and only if the specified huge integer is zero
bool HugeInt::isZero()
{
   return size == 1 && digit[ 0 ] == 0;
}
```